Session 307

Security: Keychain

Craig Keithley
Security & Cryptography Tech Manager
Worldwide Developer Relations

Session 307

Security: Keychain

Ken McLeod
Senior Engineer, Apple Data Security

Introduction

- Keychain overview
- Keychain features in Mac OS X
- Keychain Manager APIs
- Managing user interaction
- Demo and Question and Answer

Session FAQ

- What the Keychain provides for you
- How to add Keychain support to your application
- Where to get more information

Keychain Overview

- Provides a secure repository for passwords, keys, and certificates
- Built on open-source CDSA framework
- Enables clients to provide transparent authentication (single sign-on user experience)
- Available in both Mac OS 9 and Mac OS X
- Every Mac OS X user has a keychain

Keychain Terminology

- Keychain = software, keychain = data store
- A keychain is either locked or unlocked
  - Data can only be retrieved from an unlocked keychain
  - When locked, no key to decrypt the data is available
- Each stored password, key, or certificate is a keychain item
- Default keychain is the one to which new keychain items are added

Keychain Usage

- A system can have multiple keychains (typically one per user)
- More than one keychain may be unlocked
- Keychains can be set to lock automatically
- Default keychain is unlocked at login
  - Mac OS X, or Mac OS 9 with multiple users
  - Same name and password as login account

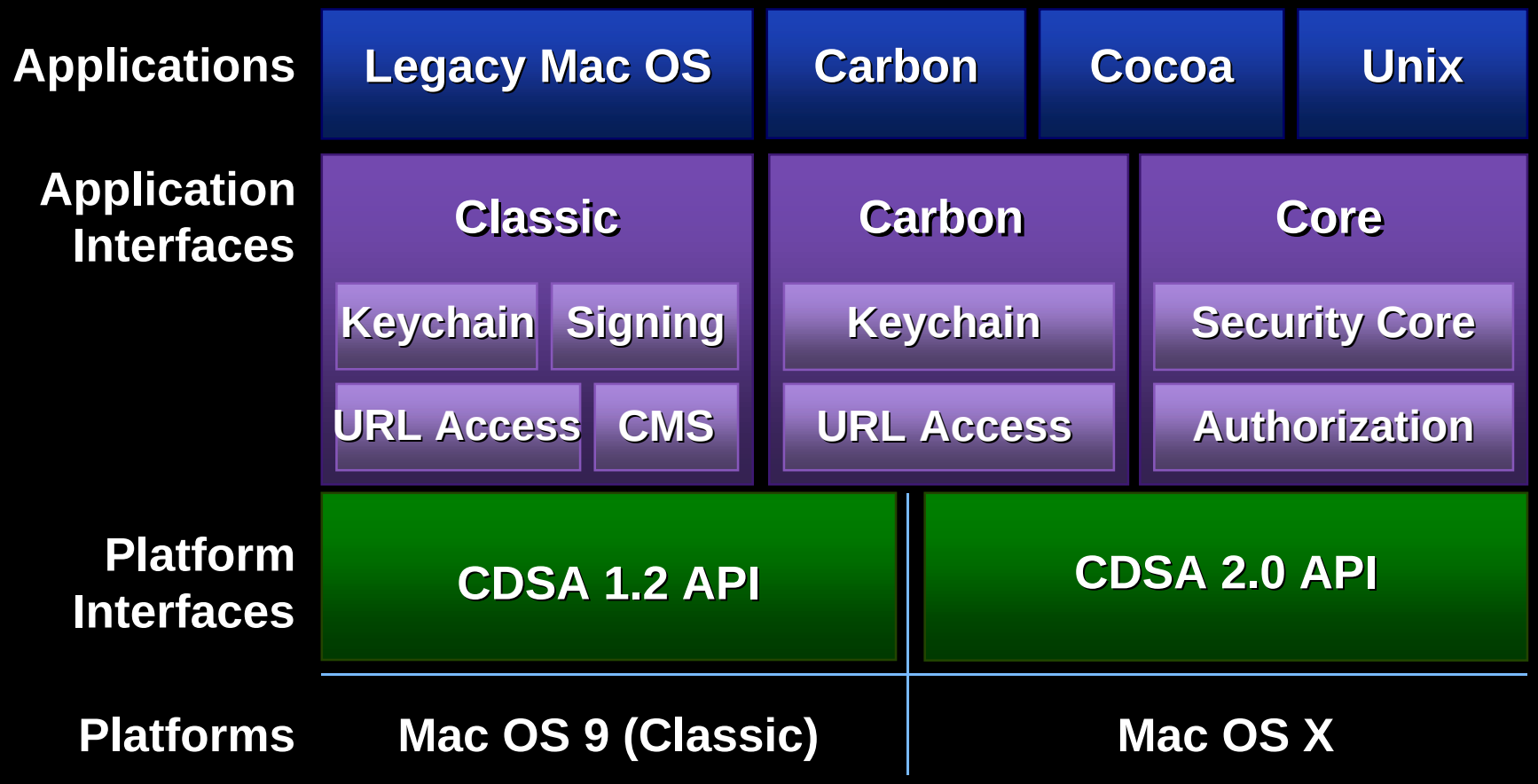
Keychain Usage (Cont.)

- Keychain files are portable to other machines
- Mac OS 9 and Mac OS X keychains are not interchangeable
- Keychain files can be located on servers and locked media
- Architecture supports keychains on Smartcards and removable tokens

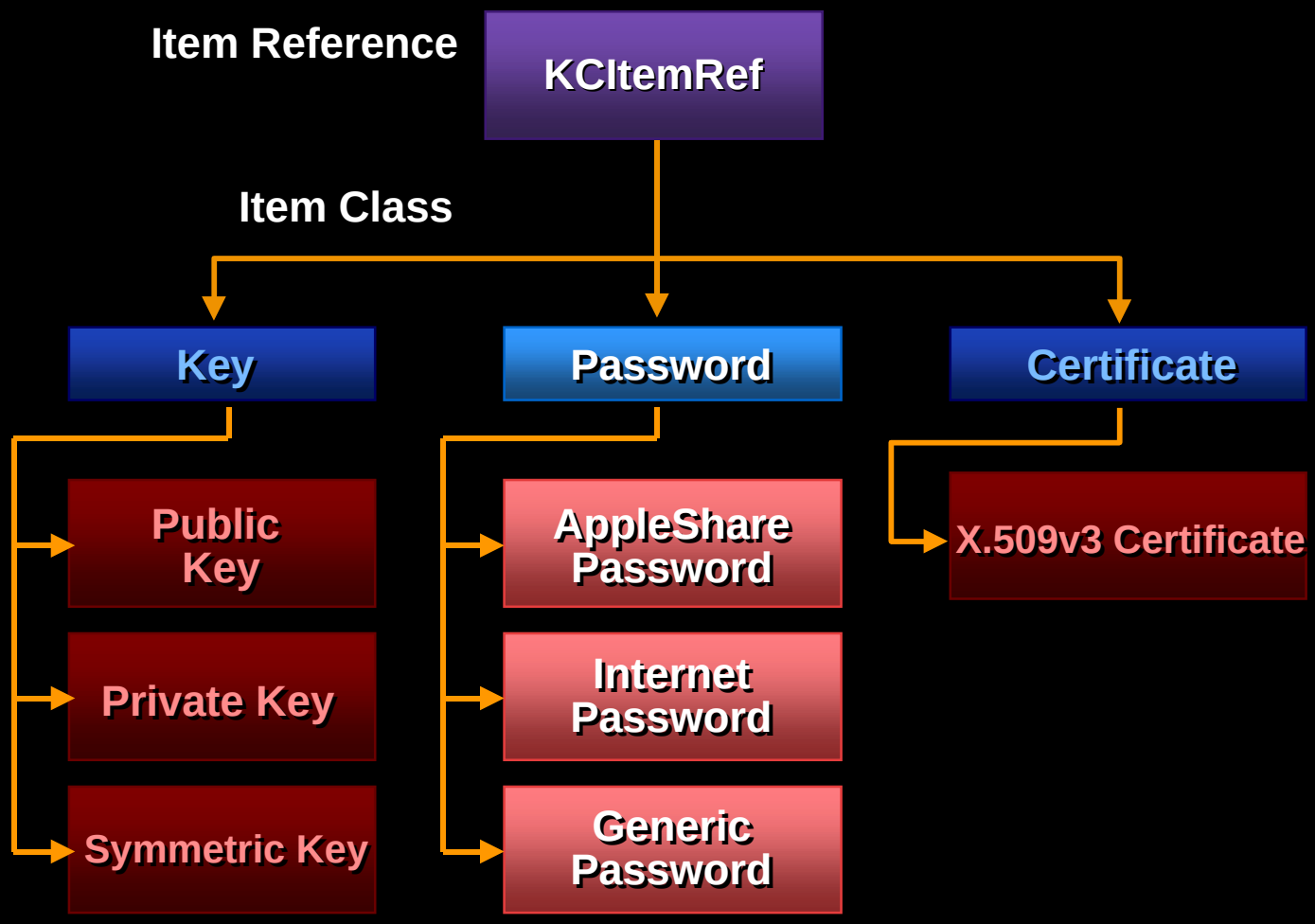
Keychain in Mac OS X

- Keychain is an integral OS feature
- Used for password storage by AppleShare, iTools, Mail, etc.
- Allows users to store arbitrary private data
- Persistent access control on a per-item basis
- Keychain API in Carbon and CoreServices frameworks

Apple Security APIs



Keychain Items



What's in a Keychain Item?

- Two components: attributes and data

Attributes	Account	"ken"
	Server	"ftp.apple.com"
	Path	"/private/stuff"
	Protocol	"ftp"
Data	"Thinkdifferent"	

Item Attributes

- All items have common attributes
  - Item class (e.g., “Internet password”)
  - Date created and modified
  - Label, description, and comment
- Each item class has additional attributes
- Attributes are used to find an item

Protecting Item Data

- Cannot be searched or used to retrieve an item
- Cannot be retrieved unless the keychain is unlocked
- Cannot be retrieved without the user's explicit permission
- Private and symmetric key data cannot be retrieved

How Secure is it?

- Doesn’t depend on machine's physical security
  - No need to write passwords down or store them insecurely on disk
  - Keychain password is never stored on disk
- Symmetric encryption key is derived from the keychain password using PBKDF2 (PKCS#5)
  - Mac OS 9: 128-bit RC2 encryption
  - Mac OS X: 168-bit 3DES encryption

Access Control

- Data of keychain items cannot be retrieved without the user's explicit permission
  - Prevents rogue applications from stealing passwords while a keychain is unlocked
  - Creating application may access the item by default

Access Control (Cont.)

- Each keychain item may have a different access control list (ACL)
- Access can be preconfigured for applications, or set “on the fly” when an item is used
- Access control lists are persistent in Mac OS X

What Can You Do with it?

- Store your application's passwords securely in a keychain instead of a preferences file
- Retrieve a password from the keychain to enable a transparent authentication experience for the user
- Store other private data (keys, cookies, PINs, credit card numbers, etc.)

API Overview

- Getting started with high-level calls
  - Finding or adding a password
- Low-level “building block” routines
  - Searching for one or more items
  - Managing keychains
  - Keychain event notification
- Carbon vs. Core Services

Getting Started

- Call **KeychainManagerAvailable**
- No need to explicitly create or unlock a keychain; let the user do it!
  - User's default keychain is already unlocked at login
- Find or add a password using high-level calls

Finding an Item

- Use high-level calls to find a single password that matches a given set of attributes

KCFindGenericPassword
KCFindInternetPassword
KCFindAppleSharePassword

Adding an Item

- Add a single password to the default keychain using high-level calls

KCAddGenericPassword
KCAddInternetPassword
KCAddAppleSharePassword

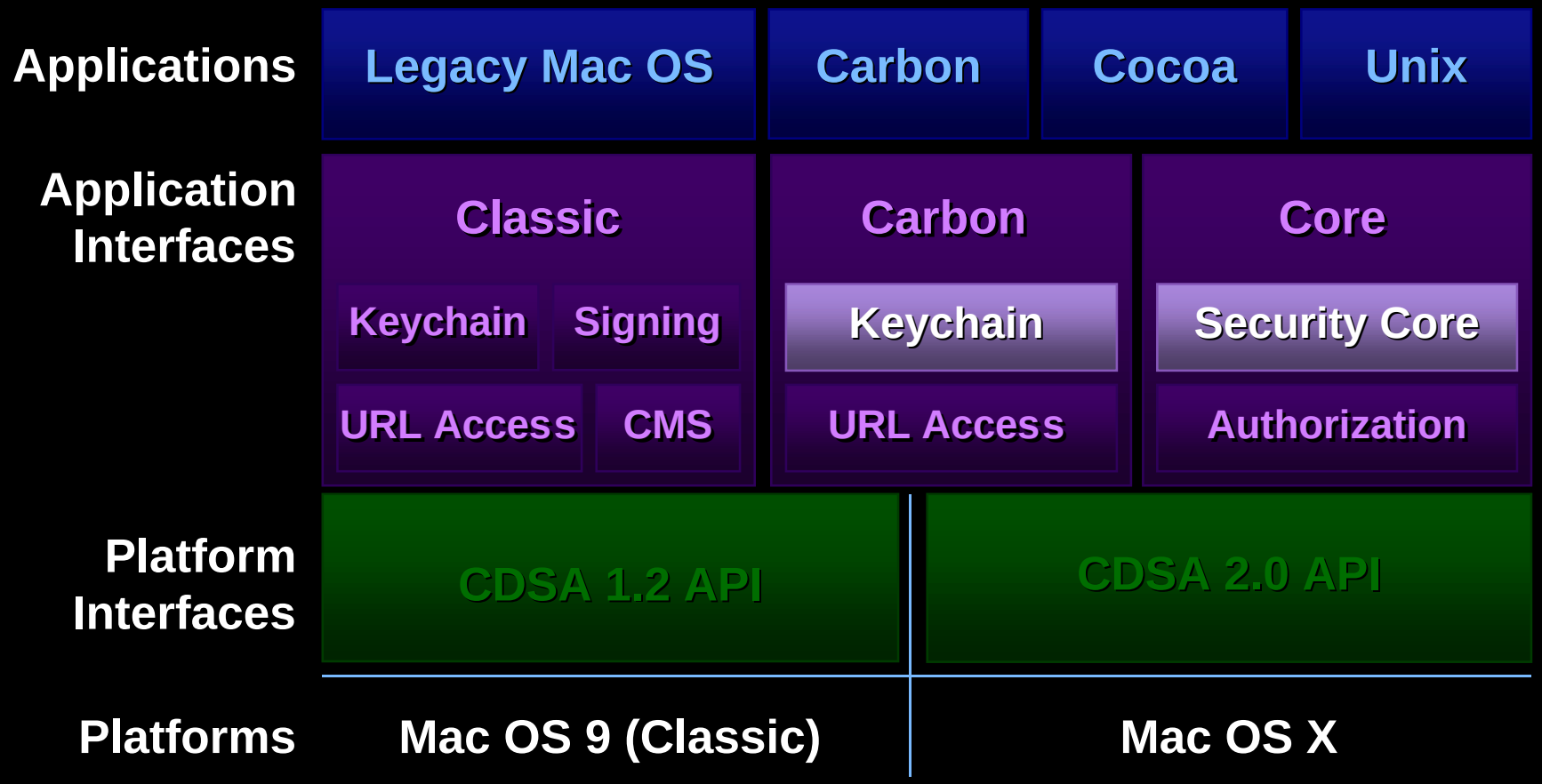
What High-Level Calls Do

- High-level routines (add, find) are implemented using low-level "building block" calls

KCAddInternetPassword =

KCNewItem
KCAddItem
KCSetAttribute
KCSetAttribute
KCSetAttribute
KCSetAttribute
KCSetAttribute
KCSetAttribute
KCSetData
KCUpdateItem

Apple Security APIs



Carbon vs. Core Services

- High-level calls are in Carbon
  - May display or require UI
  - #include <Carbon/Carbon.h>
  - Automatically includes Core Services
- Low-level calls are in Core Services
  - Do not display or require UI
  - #include <CoreServices/CoreServices.h>
  - Can be used by command-line tools or applications

Keychain Items

- Items are created by **KCNewItem**
- Items are added to the keychain by **KCAddItem**
- Items are accessed through a **KCItemRef**
- Manipulate item attributes using **KCGetAttribute** and **KCSetAttribute**
- Manipulate item data using **KCGetData** and **KCSetData**

Keychain Items (Cont.)

- Commit changes to an item using **KCUpdateItem**
- Release the item using **KCReleaseItem** when finished

Searching for Items

- Use **KCFindFirstItem** when you expect or want to find more than one item
- Continue searching with **KCFindNextItem**
- Release the search criteria when finished by calling **KCReleaseSearch**

Identifying Keychains

- KCRef identifies a keychain by reference
  - **KCMakeKCRefFromAlias**
  - **KCMakeAliasFromKCRef**
- Don't assume keychains are files or have a FSSpec/FSRef!
- Getting status information
  - **KCGetStatus**
  - **KCGetKeychainName**

Managing Keychains

- Unlocking a keychain
  - **KCUnlockKeychain**
- Creating a new keychain
  - **KCCreateKeychain**
- Iterating through all keychains
  - **KCCountKeychains**
  - **KCGetIndKeychain**

Notification

- Keychain events
  - Unlock, lock, add, find, get data, delete, update, default changed
- If you want to be notified of keychain events, install a callback with **KCAddCallback**
- Register only for events you care about
- Use **KCRemoveCallback** when done

Sample Code

```
ConnectToServer (StringPtr server, StringPtr account)
{
    status = KCFindInternetPassword (
                        server,             // “www.apple.com”
                        nil,                // securityDomain
                        account,            // “ken”
                        kAnyPort,
                        kKCProtocolTypeHTTP,
                        kKCAuthTypeDefault,
                        bufferLength,
                        &buffer,
                        &actualLength,
                        nil );

    if (status != noErr ) MyGetPassword (&buffer, &actualLength);
    Connect (server, account, buffer);
}
```

Platform API Support

- Dual-platform Carbon application (CFM):
  - Link with CarbonLib (1.1 or later) on Mac OS 9
- Carbon application:
  - Link with Carbon framework on Mac OS X
- Cocoa application:
  - Link with either Carbon or CoreServices framework
- Unix command-line application:
  - Link only with CoreServices framework

Managing User Interaction

- Problem: what if no user interaction can occur?
  - Call **KCSetInteractionAllowed** (Boolean state); only useful if keychain is already unlocked
  - Without UI, keychain password must be provided programmatically to **KCUnlock**
- Let the user decide when to interact
  - Provide a “Remember Password” option
  - API calls should be direct result of a user action

Managing Keychain Data

- Cache retrieved password items when appropriate (e.g. polling a POP mail account)
- Lock memory containing sensitive data you've retrieved from a keychain
  - **HoldMemory and UnholdMemory in Carbon**
  - **mlock and munlock on Mac OS X (4.4 BSD)**

Demo

Summary

- Keychain is a secure repository for passwords, keys, and digital certificates, provided for you as a system service
- Easy to use; only a few calls are needed for most applications
- Get started now with Mac OS X!

Resources

Keychain Documentation
/Developer/Documentation/Carbon/securityservices/

Security
Specifications and SDKs
http://developer.apple.com/macos/security.html

CDSA 2.0
Specifications
http://www.opengroup.org

Related Sessions

305 Security: Overview An overview of security on Mac OS X	Room A1 **Wed., 9:00am**
306 Authorization APIs Interfaces for HTTP and FTP	Room J2 **Wed., 10:30am**
309 Security: Kerberos Learn about Kerberos	Room A2 **Thurs., 2:00pm**
FF113 Feedback Forum: Security Talk to us	Room C **Fri., 10:30am**

Who to Contact

Craig Keithley
Apple Worldwide Developer Relations,
Security and Cryptography Technology Manager
keithley@apple.com

Q&A

Craig Keithley
Security and Cryptography Technology Manager
keithley@apple.com

http://developer.apple.com/events/urls.html

WWDC

Worldwide Developers Conference 2001